CS 189: Introduction to Machine Learning
Spring 2014
Homework 1: Digit Classification

Due: 11:59 pm on Friday, January 31, 2014

Introduction

In this assignment, we will do digit recognition using the original MNIST dataset. The state-of-the-art error rate on this dataset is roughly 0.5%; the methods in this assignment will get you to around 8% (modulo implementation details). We will try to get closer to the state-of-the-art as we move further in the course.

Before starting, please read Appendix B. To avoid computational expense, we will only train on a subset of the data. (Once you have finished this assignment, if you have enough computing power at your disposal, you can try your code on the full dataset.) The file **train_small.mat** contains 7 different subsets, containing 100, 200, 500, 1000, 2000, 5000 and 10000 training points respectively. The full training set containing 60000 images is in **train.mat**. The test set is in **test.mat**.

We encourage you to work in groups of **at most 2**.

1 Support Vector Machines

We will use linear support vector machines to classify handwritten digits from 0–9. In this assignment, we will use the simplest of features for classification: pixel brightness values. Simply put, our feature vector for an image would be a row vector with all the pixel values concatenated in a row major (or column major) format.

Problem 1
Train a linear SVM using raw pixels as features. Plot the error rate on the test set versus the number of training examples. State your choice of parameters for training the SVM.

2 Confusion Matrix

In a multiclass classification setting, a confusion matrix is normally used to report the performance of your algorithm. Briefly, a confusion matrix is a matrix whose rows correspond to the actual class and columns indicate the predicted class. A more detailed explanation can be found at **http://en.wikipedia.org/wiki/Confusion_matrix**

Problem 2
Create confusion matrices for your experiments in Problem 1. Color code your results and report them. What insights can you get about the performance of your algorithm from looking at the confusion matrix?

3 Cross-validation

A common practice while performing machine learning experiments is to perform cross-validation to select model parameters. In this assignment, we will use k-fold cross-validation to determine a good value for the SVM regularization parameter **'C'**.

Briefly, cross-validation is a technique to assess how well your model generalizes to unseen data. In k-fold cross-validation, the training data is randomly partitioned into k disjoint sets and the model is trained on $k-1$ sets and tested on the k^{th} set. This process is repeated k times with each set chosen as the test set once. The cross-validation accuracy is reported as the average accuracy of the k iterations.

While trying to fix a model parameter, the above process is repeated for a number of different values of the parameter, and the parameter value with the highest cross-validation accuracy is selected. The final trained model is obtained by training on the entire training set with the obtained parameter value. There are other forms of cross-validation too (**http://en.wikipedia.org/wiki/Cross-validation_(statistics)**).

Problem 3
Explain why cross-validation helps. Find the optimal value of the parameter **'C'** using 10-fold cross-validation on the training set with 10,000 examples. Train an SVM with this value of **'C'** and report the test error rate.

You should not use LIBLINEAR's built-in cross-validation functionality; you should implement it yourself.

Appendix

A - Deliverables

You should submit a zip file that contains the following three things:

- Your code. We should be able to run it out of the box.
- A short write-up containing the answers to the above. The write-up should be a PDF with your images and plots inside.
- A short README that contains (1) your name and student ID, (2) the name and student ID of your partner, if you have one, and (3) instructions on how to use your code to reproduce your results.

We will announce submission instructions on Piazza later before the deadline.

B - Setting up MATLAB and LIBLINEAR

Before you begin, you need to make sure everything compiles well. You will need MATLAB, and either a Mac or Linux. If you are in Windows, you will need a version of make. Most files are just MATLAB, so no compilation is required. However, you need to compile the LIBLINEAR package that you will download from `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`. Place the directory you download from LIBLINEAR into the code subdirectory and cd into it, e.g cd tocode/liblinear-x, then type make. Then cd into code/liblinear-x/matlab. Edit the Makefile to point to your local copy of MATLAB, and then type make. For further information, please read the README file in the package.

The MNIST dataset is provided in the code folder. We provide two functions for you. `benchmark.m` will take in the predicted labels and true labels and return the error rate, and also the indices of the labels that are wrong. `montage_images.m` will produce a montage of a set of images: use this, for instance, to visualize the images on which you make errors.